7. Prototípus koncepciója

35 – negalt

Konzulens:
Ludmány Balázs

Csapattagok

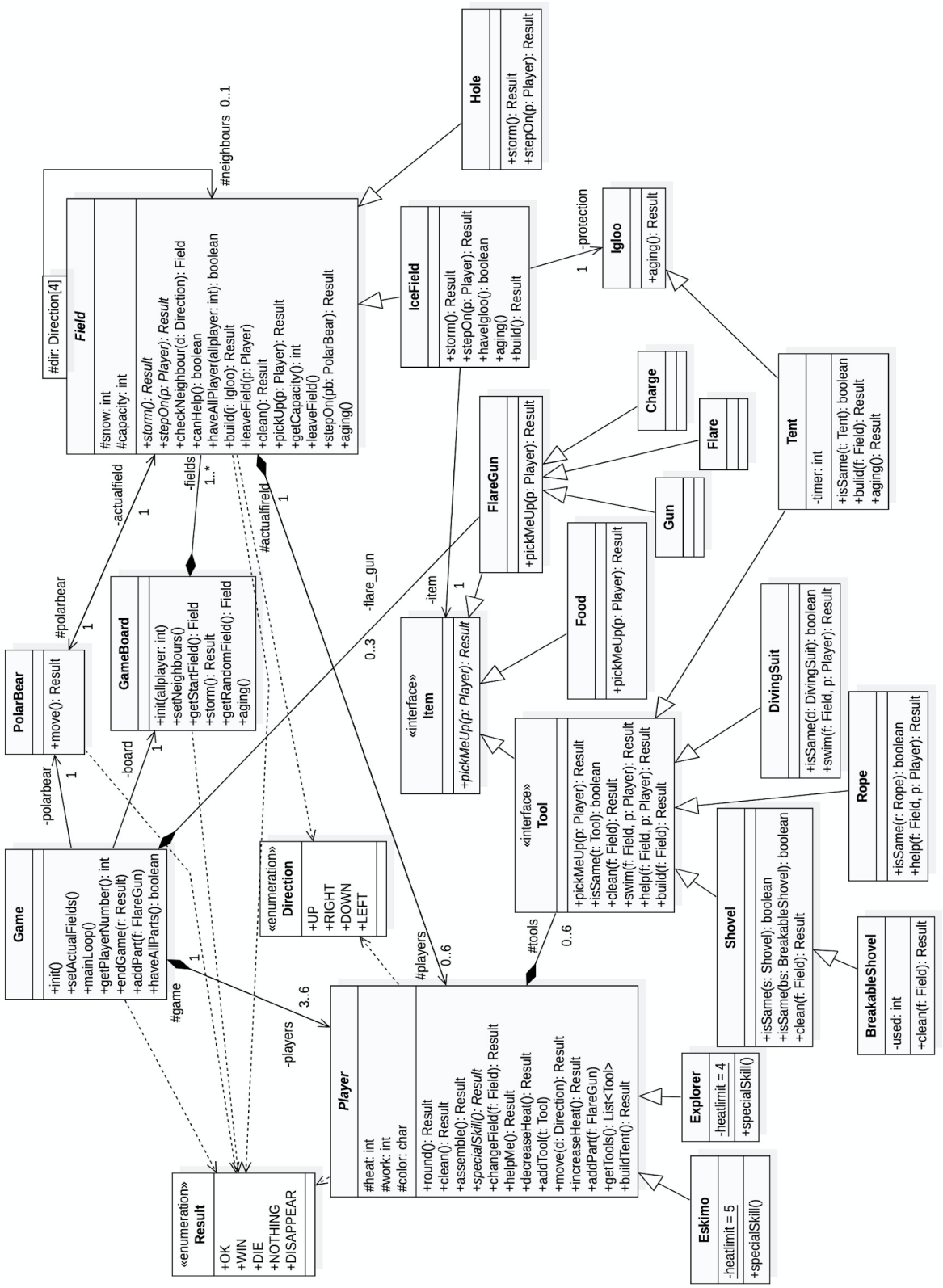
Fábián Csenge Zita	KGM10V	csengefabian@gmail.com
Gutai Auguszta Matild	QWG59W	auguszta123@gmail.com
Ilosvay Viktória Brigitta	M27F3W	ilosviki@gmail.com
Szabó Kinga Johanna	G003M6	szabo.kinga1999@gmail.com
Szabó Marcell	LAUPQQ	szabomarcell.usa@gmail.com

2020-04-01

7. Prototípus koncepciója

7.0 Változás hatása a modellre

7.0.1 Módosult osztálydiagram



7.0.2 Új vagy megváltozó attribútumok, metódusok

BreakableShovel osztály bevezetése:

Felelősség:

- Törékeny ásó kezelésére szolgáló osztály.

Ősosztályok:

- Shovel → BreakableShovel

Attribútum:

- **int used**: Reprezentálja, hogy hányszor használták már az ásót. Alapértéke mindig 3, és amennyiben 0-ra csökken az értéke, akkor eltörik és megsemmisül.

Metódus:

- **Result clean (Field f)**: Akkor hívódik meg, ha az ásást végző játékosnál van törékeny ásó. Ekkor a függvény meghívja a *Field* osztály clean() metódusát, ezzel megpróbálva még egy réteget letakarítani a mezőről. Amennyiben ez NOTHING-gal tér vissza, akkor a ő is NOTHING-gal fog. Ha ez OK-kal tér vissza, akkor a clean(Field) csökkenti eggyel a used attribútumát, majd ellenőrzi, hogy a used értéke 0-ra csökkent-e. Abban az esetben ha igen, akkor DISAPPEAR-rel tér vissza, különben pedig OK-kal.

Eskimo osztály módosított metódusa:

- **void specialSkill()**: A Player osztályban lévő absztrakt függvény megvalósítása. Létrehoz egy Igloo objektumot, majd ezt átadva hívja meg a *Field* osztály build(Igloo) függvényét. Végül ennek visszatérési értékével (OK/NOTHING) tér vissza ez a metódus is.

***Field* osztály módosításai:**

Attribútumok:

- **Item item**: Átkerült az analízismodellünkben az IceField osztályhoz.
- **protected PolarBear polarbear**: A játékban szereplő medvét tárolja el.

Metódusok:

- **void leaveField()**: A polarbear attribútumát NULL-ra állítja.
- **Result stepOn(PolarBear pb)**: A polarbear attribútumát a megkapott PolarBear példányra állítja. Ezt követően megvizsgálja, hogy az adott mezőn van e igloo. Amennyiben igen, akkor a függvény OK értékkel tér vissza, mivel ha van is ott játékos, az igloo akkor is megvédi az ott lévőket. Ellenkező esetben, tehát ha a mezőn nincs igloo, akkor pedig a mezőn álló játékosok száma kerül vizsgálatra. Ha áll (tehát a szám nagyobb mint 0), akkor a jegesmedve elkapja az ott lévőket, így a metódusnak DIE, különben pedig OK visszatérési értéke lesz.
- **virtual Result build(Igloo i)**: Az eddig buildIgloo() függvény törlésre került az analízis modellünkben, és helyette lett ez a módosított függvény, amely egy üres virtuális metódus. Nem csinál semmit sem, csak visszatér egy OK-kal.
- **virtual void aging()**: Virtuális üres függvény.

Game osztály módosításai:

Attribútum:

- **Polarbear polarbear**: A játékban szereplő medvét tárolja el.

Metódusok:

- **void init()**: A játék kezdetekor bekéri a játékosok számát majd sorra azoknak a karaktertípusát. Létrehozza a GameBoard-ot, majd meghívja az osztály init(int) metódusát átadva neki a játékosok számát. Ezt követően a GameBoard getStartField() metódusának segítségével lekéri a játékosok kiindulási mezőjét (bal felső). Majd a bekért adatok alapján konstruktorukban átadva a kiindulási mezőjüket létrehozza az eszkimókat, illetve a sarkkutatókat reprezentáló osztályokat. Végül a GameBoard osztály getRandomField() függvényének visszatérését átadva konstruktorban létrehozza a Jegesmedvét, aminek az így kapott mező lesz az actualfield-je.

- **void setActualFields()**: Ez a metódus kikerült az analízismodellből.

GameBoard osztály módosított metódusai:

- **Field getRandomField()**: Visszaad egy random mezőt, ami nem lehet a bal felső sarokban található 4 mező egyike sem.
- **void aging()**: Meghívja minden egyes Field-re az aging() függvényt.

Hole osztály módosított metódusa:

- **Result stepOn(Player p)**: A *Field* osztályban lévő absztrakt függvény megvalósítása. Legelőször megnézi, hogy tartózkodik-e az adott mezőn jegesmedve, mivel ha igen, akkor a függvény DIE-al tér vissza. Ha ezt a vizsgálatot követően nem keletkezett visszatérési érték, akkor az attribútumként kapott Player példány helpMe() metódusát hívja meg, majd ennek a visszatérési értékével tér vissza a stepOn(Player) függvény is.

IceField osztály módosításai:

Attribútumok:

- **Igloo protection**: Az analízismodellünkből törlődik az igloo attribútum, és helyette lesz a protection, amely megadja, hogy az adott jégmezőre (icefield-re) van-e építve bármi.
- **Item item:** Tárolja, hogy a mezőn milyen tárgyat lehet felvenni.

Metódusok:

- **Result stepOn(Player p)**: A *Field* osztályban lévő absztrakt függvény megvalósítása. Legelőször megnézi, hogy tartózkodik-e az adott mezőn jegesmedve, mivel ha igen, akkor a függvény DIE-al tér vissza. Ha ezt a vizsgálatot követően nem keletkezett visszatérési érték, akkor az adott mezőn lévő játékosok száma kerül vizsgálatra. Amennyiben elbírja az ott tartózkodó karaktereket a jégtábla, akkor OK, ellenkező esetben pedig DIE értékkel fog visszatérni a metódus.
- **Result build(Igloo i)**: Az eddig buildIgloo() függvény törlésre került az analízis modellünkben, és helyette lett ez a módosított függvény, amely A *Field* osztály virtuális build(Igloo) metódusának a felüldefiniálása. Megvizsgálja, hogy az adott mezőn található-e már építmény (sátor vagy igloo). Ha igen, akkor visszatér NOTHING-gal, különben pedig OK-kal.
- **void aging()**: Ha az IceField osztály protection attribútuma nem NULL, akkor meghívja rá a az Igloo aging() függvényét. Ha ez DISAPPEAR-rel tér vissza, akkor törli a protection attribútumát (NULL-ra állítja).

Igloo osztály bevezetése:

Felelősség:

- Igloo és sátor megkülönböztetésére szolgáló osztály.

Metódus:

- **Result aging()**: Virtuális üres függvény, ami csak visszatér OK-kal.

Player osztály módosításai:

Attribútum:

- **char color:** Tárolja a játékos színét.

Módosított metódusok:

- **Result buildTent()**: Meghívja a tools attribútumban tárolt összes Tool példányra a build(Field) függvényt. Amennyiben bármelyik DISAPPEAR-rel ré vissza, akkor az törlésre kerül a tools tömbből, majd a buildTent() metódus visszatérési értéke OK lesz. Abban az esetben viszont, ha egyik sem tér vissza DISAPPEAR-rel, akkor a buildTent() OK helyett NOTHING visszatérési értéket fog adni.
- **Result clean()**: Meghívja a *Field* osztály clean() metódusát. Amennyiben ez OK-kal tér vissza, akkor meghívja a tools attribútumban tárolt összes Tool példány clean(Field) függvényét. Ha ezekből valamelyik DISAPPEAR-rel tér vissza, akkor

azt törli a tools tömbből. Végül visszatér azzal, amivel a *Field* osztály először meghívott clean() függvénye tért vissza.

PolarBear osztály bevezetése:

Felelősség:

- A medve mozgásáért felelős osztály.

Attribútum:

- **actualfield**: Tárolja, hogy a jegesmedve hol található.

Metódus:

- **Result move()**: Először addig hívogatja meg átadva neki random generált irányokat a *Field* osztály checkNeighbour(Direction) függvényét, amíg az nem NULL értékkel tér vissza. Ezt követően szintén a *Field* osztály, bár most a leaveField() függvénye kerül meghívásra. Végül pedig a stepOn(PolarBear) metódusa hívódik meg. Amivel visszatér, azzal fog a move() is.

Result osztály bővült egy literállal:

- **DISAPPEAR**: A függvény futása során a tárgy élettartama lejárt, vagy áthelyeződött más osztályhoz a tárgy (pl.: sátor megépül). Ekkor ennek tárolását meg kell szűntetni.

Shovel osztály módosított metódusai:

- **boolean isSame(BreakableShovel bs)**: Mindig TRUE értékkel tér vissza (hiszen csak akkor hívódik meg, ha egy BreakableShovel példány szeretné magát összehasonlítani vele).
- **Result clean(Field f)**: Akkor hívódik meg, ha az ásást végző játékosnál van ásó. Ekkor ez a függvény meghívja a *Field* clean() metódusát, ezzel még egy réteget ellapátolva arról (persze, ha ez lehetséges). A metódus minden esetben OK visszatérési értékkel rendelkezik.

Tent osztály bevezetése:

Felelősség:

- Sátor eltűnésének, illetve megépítsének kezelésére szolgáló osztály.

Ősosztály:

- Igloo → Tent

Interfész:

- Tool

Attribútum:

- **int timer**: A sátor felállítása óta eltelt időt reprezentálja. Megépítéskor megegyezik az értéke a játékosok számával. Ha lecsökken nullára, akkor eltűnik.

Metódus:

- **boolean isSame(Tent t):** Mindig TRUE értékkel tér vissza (hiszen csak akkor hívódik meg, ha egy Tent példány szeretné magát összehasonlítani vele).
- **Result build(Field f):** Meghívja az attribútumként kapott *Field* osztály build(Igloo) metódusát, átadva neki önmagát. Ha ez OK-kal tér vissza, akkor a build(Field) DlSAPPEAR-rel, különben pedig NOTHING-gal fog visszatérni.
- **Result aging()**: Eggyel csökkenti a timer attribútumát. Ha így nullára csökken, akkor DISAPPEAR-rel tér vissza, különben pedig OK-kal.

Tool osztály módosításai:

Felelősségváltozás:

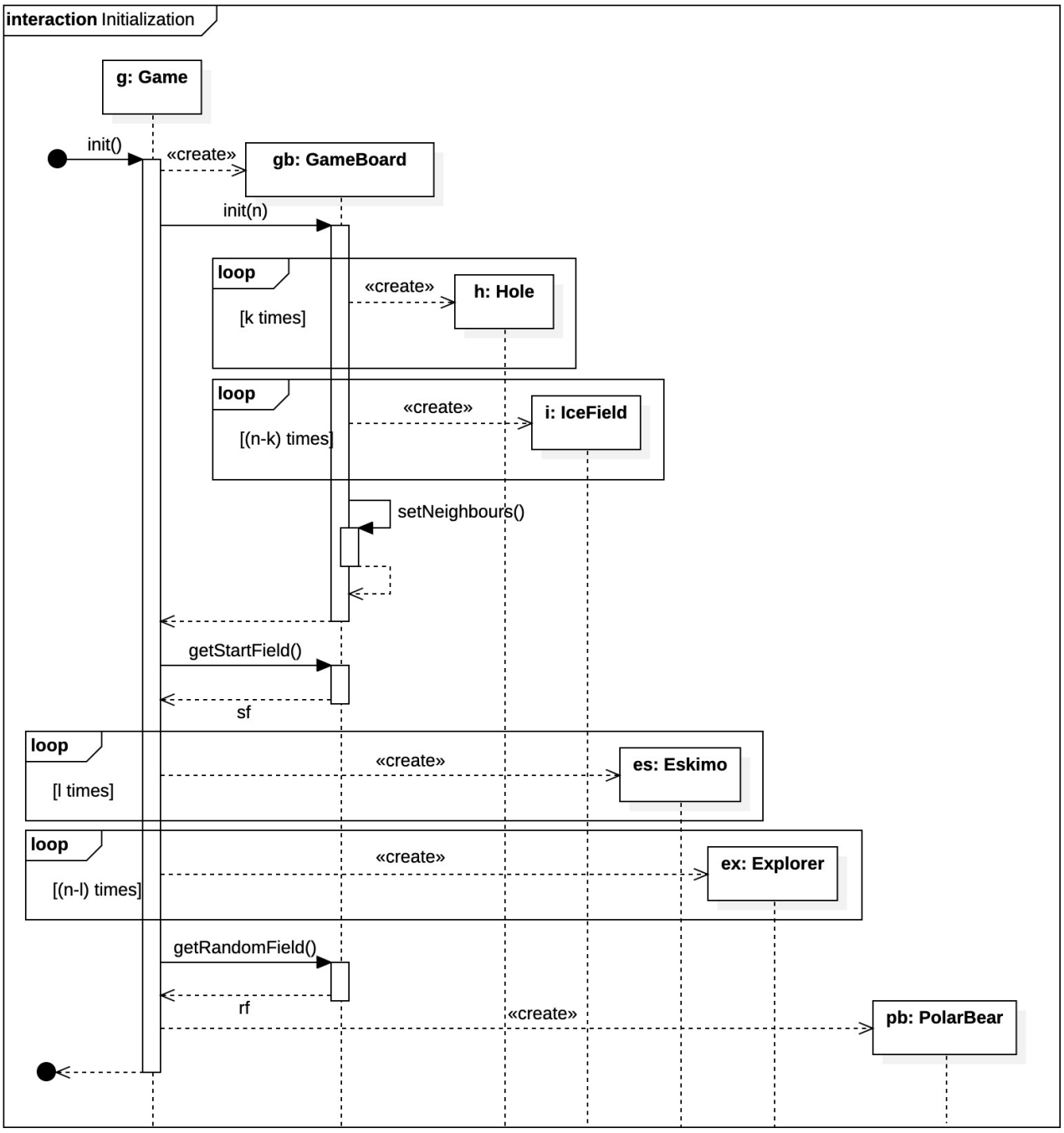
- Interfész, amely az eszközök felvételének, illetve a velük kapcsolatos interakciók (ásás, köteles kimentés, búvárruha használatával történő kimenekülés, sátorépítés) kezelésére szolgáló osztály. (Természetesen azok az osztályok, amelyek eddig a Tool osztály leszármazottai voltak, mostantól azoknak ez nem az ősosztálya, hanem az interfésze lesz.)

Módosított metódusok:

- **virtual Result build(Field f)**: Virtuális függvény, amely NOTHING értékkel tér vissza.
- **virtual Result clean(Field f)**: Virtuális függvény, amely NOTHING értékkel tér vissza.

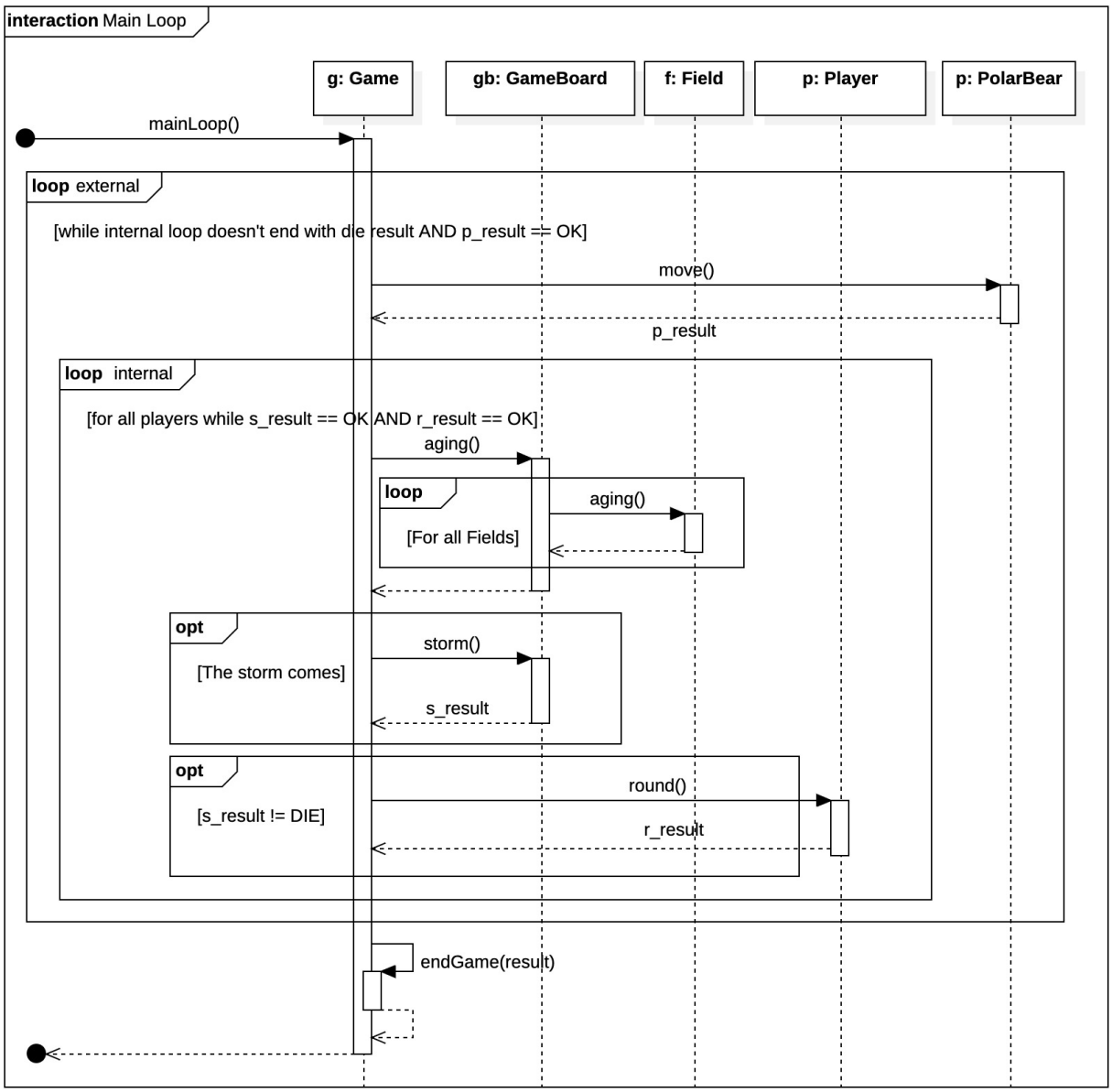
7.0.3 Szekvencia-diagramok

7.0.3.1 Initialization



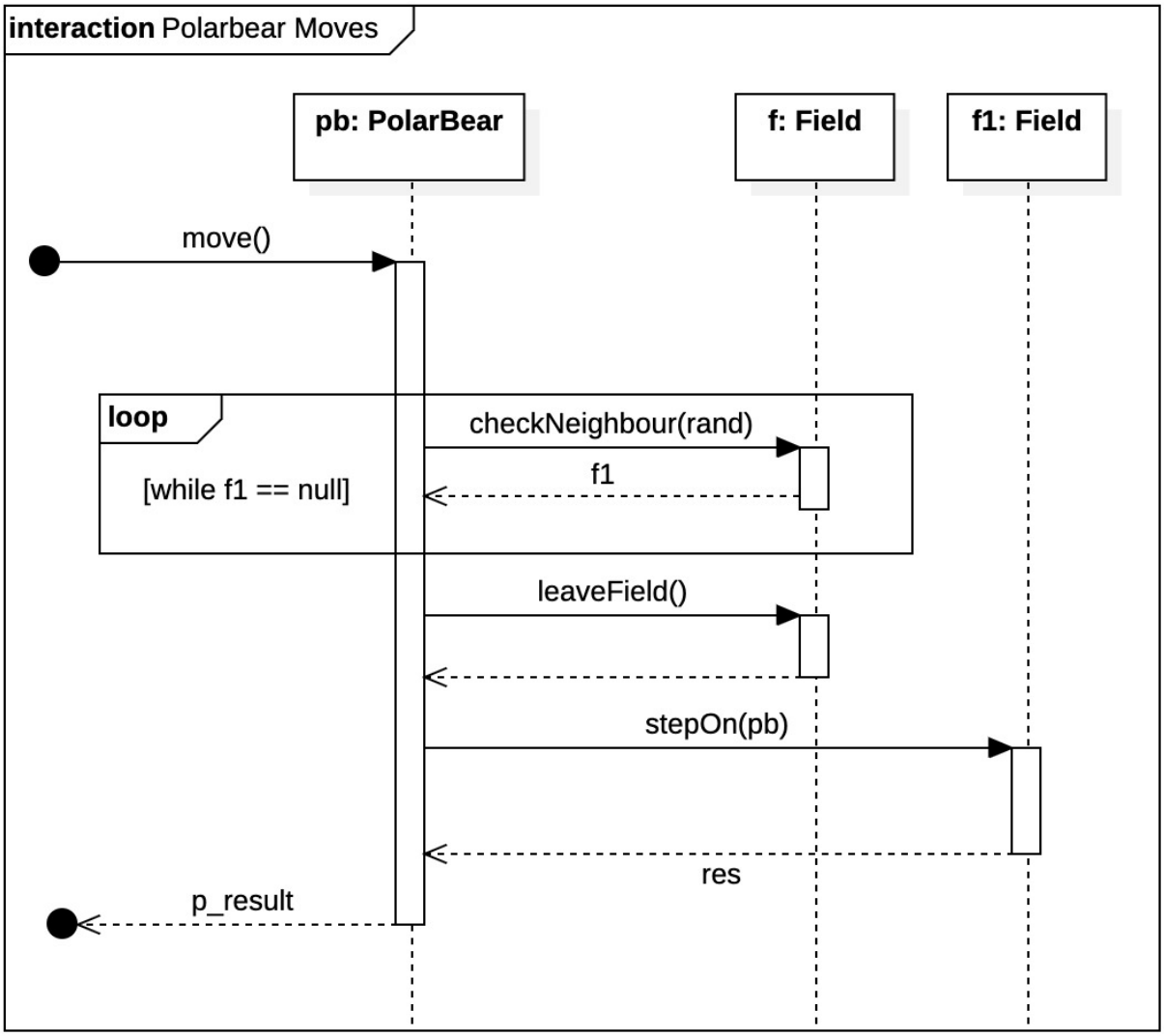
A gb példány megegyezik a Game osztály board attribútumában tárolt példánnyal, tehát ismeri.
Az n a játékosok számát adja meg.

7.0.3.2 Main Loop



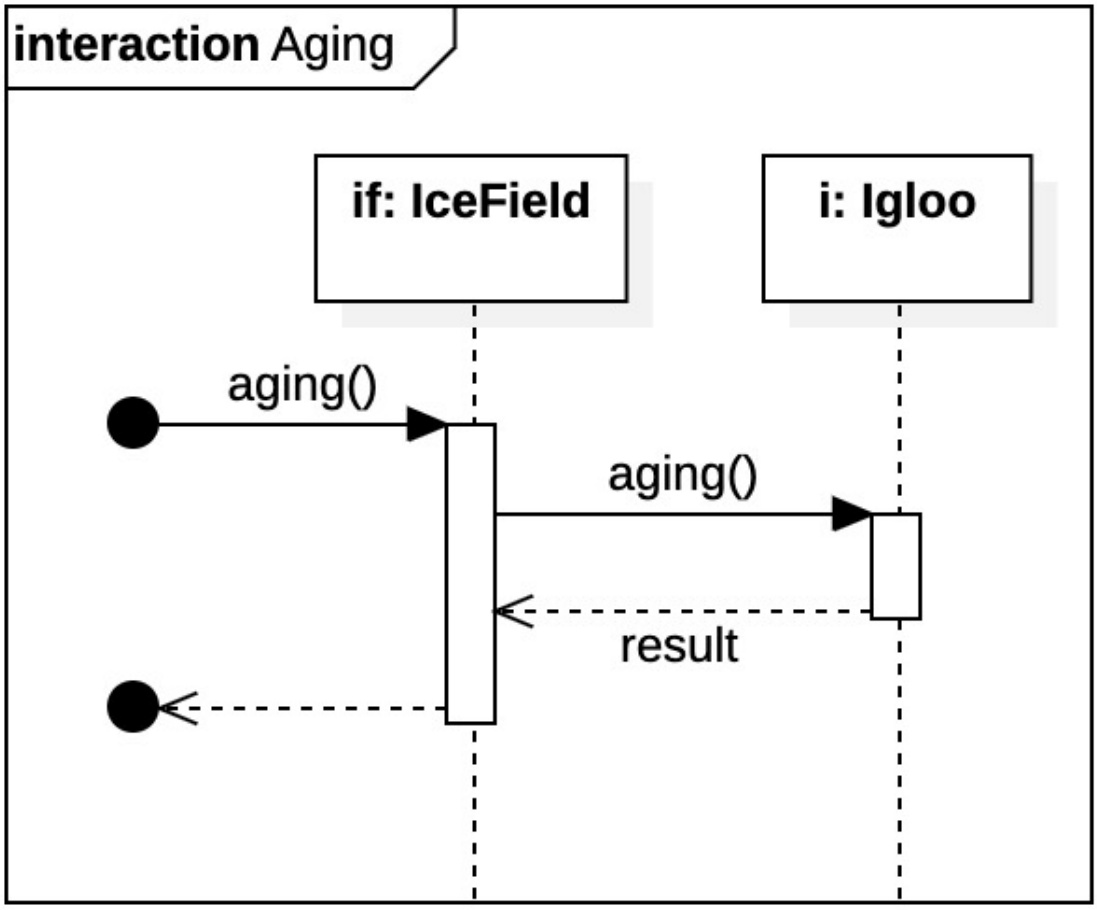
A p példány megegyezik a Game osztály polarbear attribútumában tárolt példánnyal, tehát ismeri.
A gb példány megegyezik a Game osztály board attribútumában tárolt példánnyal, tehát ismeri.
A Field példányokat a GameBoard osztály a fields attribútumában tárolja, tehát ismeri azokat.
A Player példányokat a Game osztály a players attribútumában tárolja, tehát ismeri azokat.

7.0.3.3 Polar Bear Moves



Az f példány megegyezik a PolarBear osztály actualfield attribútumában tárolt példánnyal, tehát ismeri.
A checkNeidghbour(rand) függvény visszatér az f1 referenciájával, tehát azt is ismeri.
Az f2 példány megegyezik a PolarBear osztály actualfield attribútumában tárolt példánnyal a changeField() függvény lefutását követően, tehát ismeri. Ez az f2 példány akár megegyezhet az f1 példánnyal (ha f1-en nem volt igloo, és a jegesmedvének nem kellett továbbmennie)

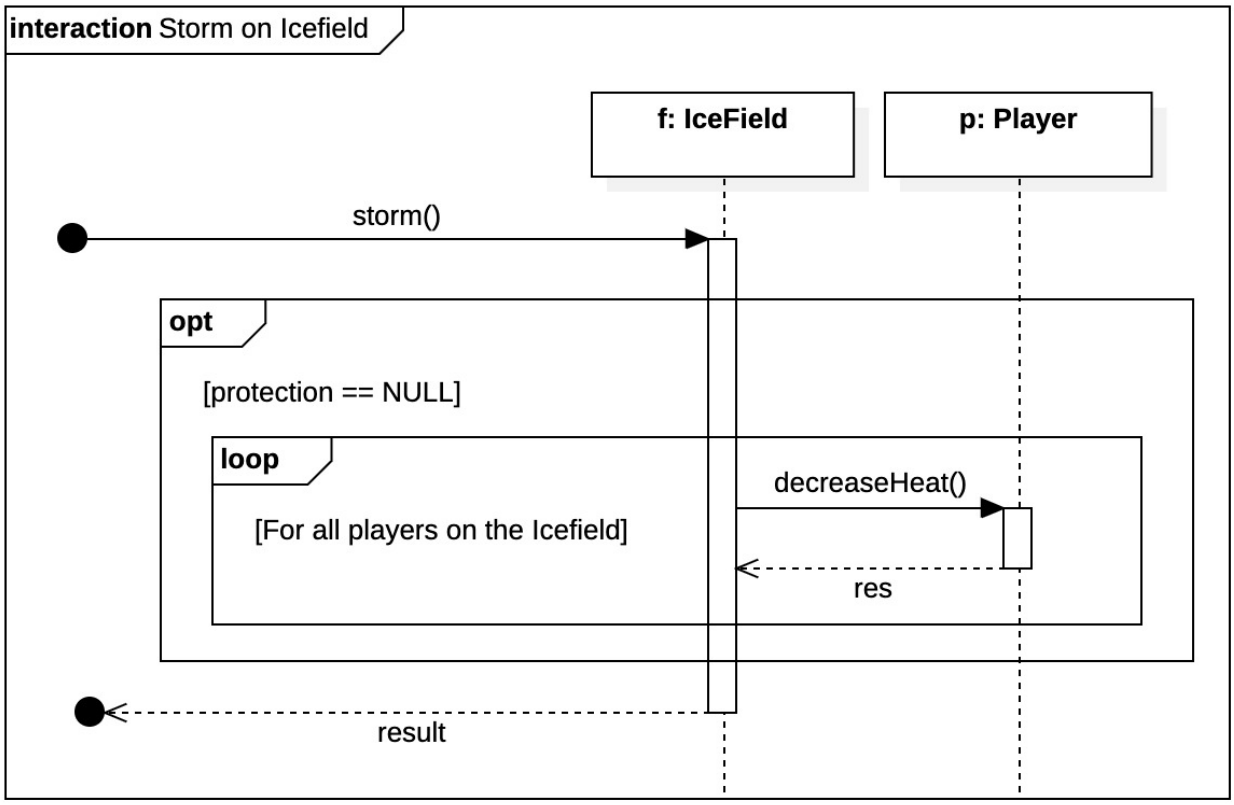
7.0.3.4 Aging



Az i példány megegyezik az IceField osztály protection attribútumában tárolt példánnyal, tehát ismeri.

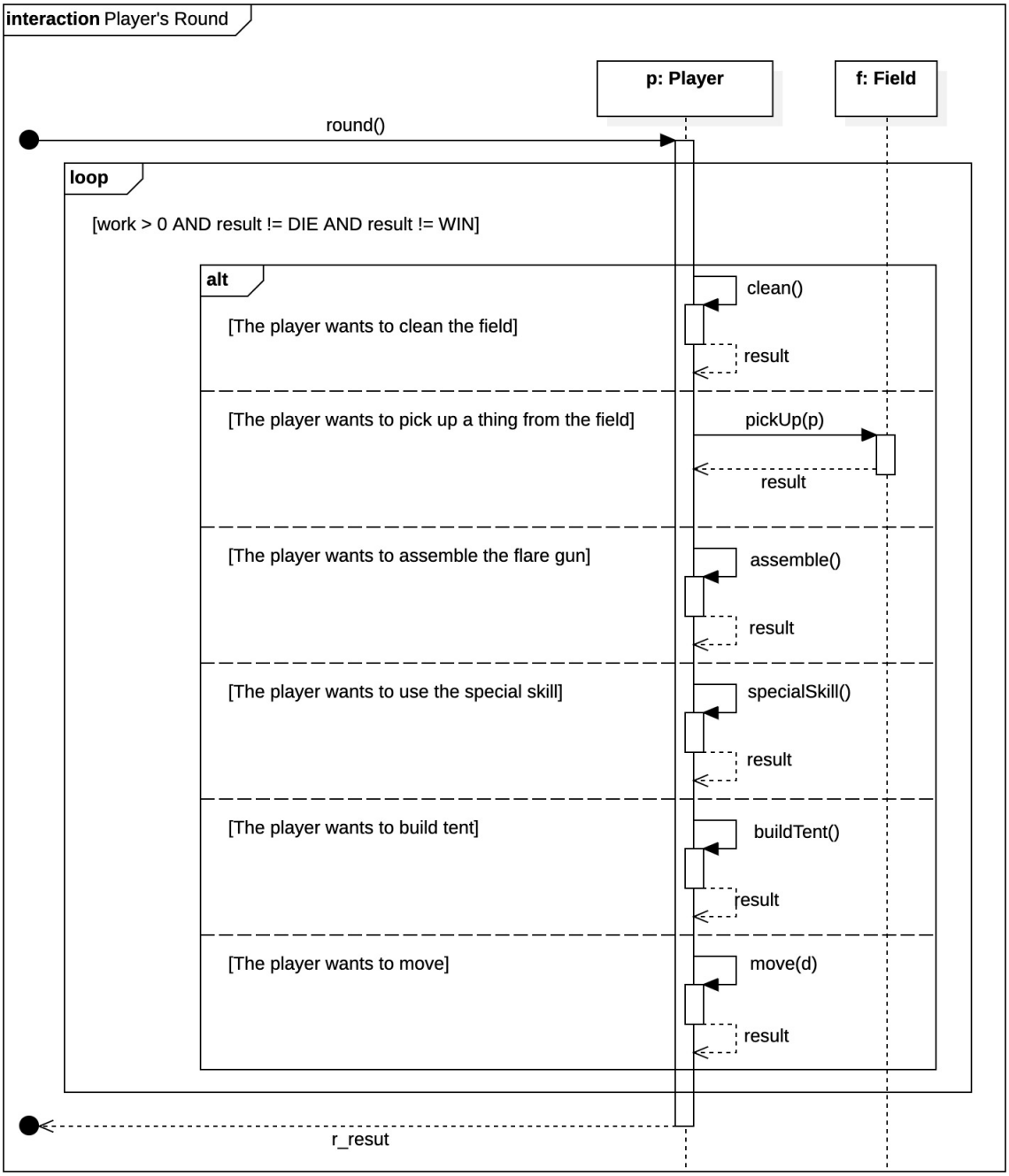
7.0.3.5 Storm on IceField

Ezen a diagramon csak az opt feltétele változott (mivel az IceField attribútumának neve és típusa megváltozott).



A player példányokat az IceField osztály a players attribútumában tárolja, tehát ismeri azokat.

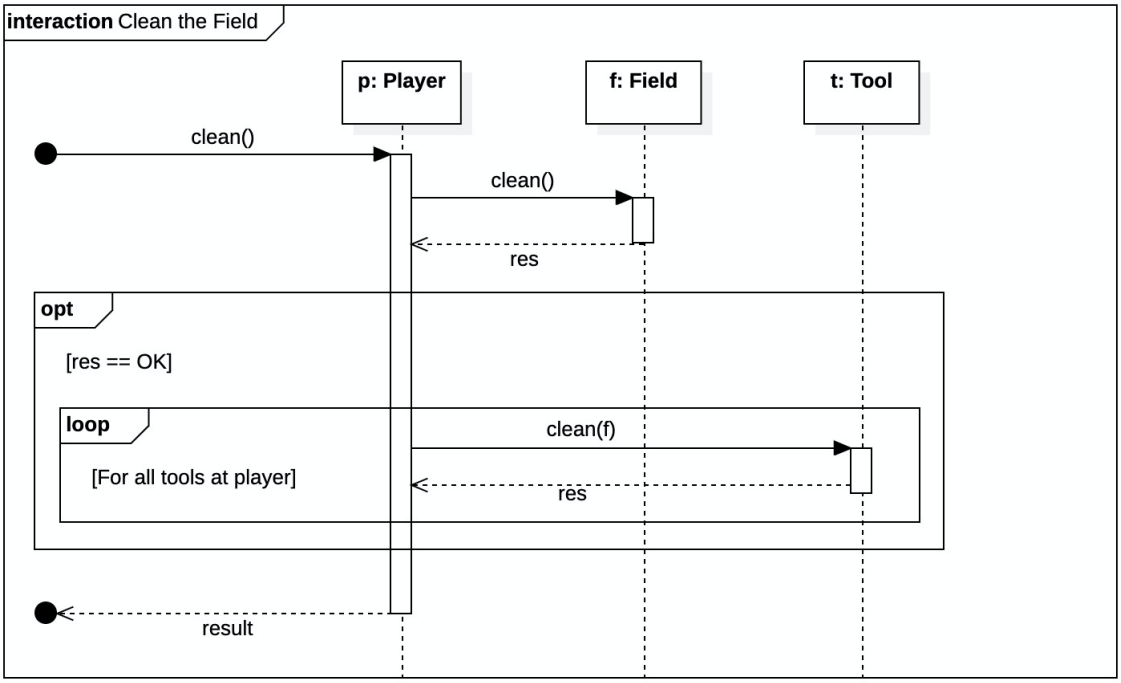
7.0.3.6 Player's Round



Az f példány megegyezik a Player osztály actualfield attribútumában tárolt példánnyal, tehát ismeri.

7.0.3.7 Clean the Field

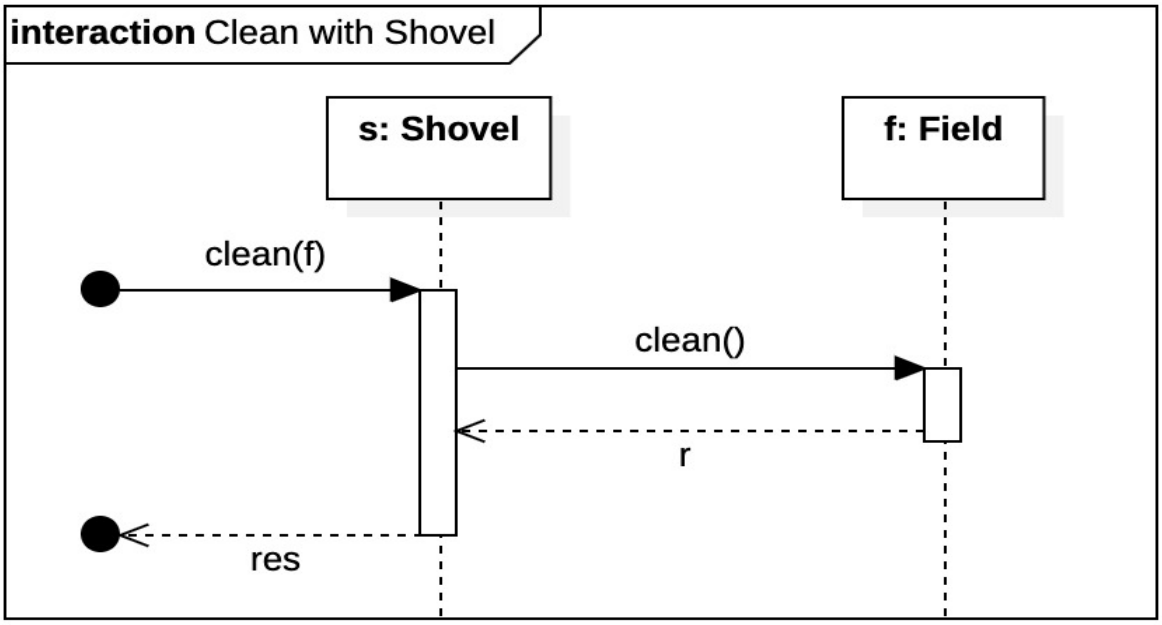
Ezen a diagramon csak az változott, hogy a Tool példány clean(f) metódusának lett visszatérési értéke



Az f példány megegyezik a Player osztály actualfield attribútumában tárolt példánnyal, tehát ismeri.
A t Tool-okat a játékos a tools attributumában tárolja.

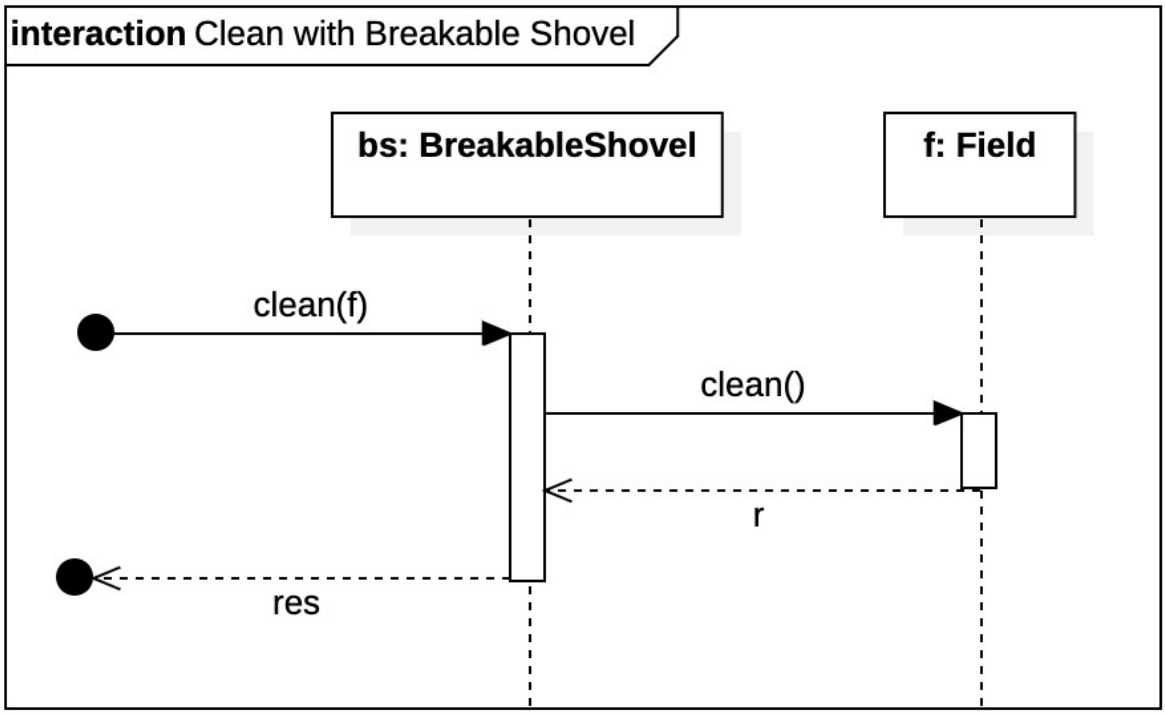
7.0.3.8 Clean with Shovel

Ezen a diagramon csak az változott, hogy a Shovel példány clean(f) metódusának lett visszatérési értéke



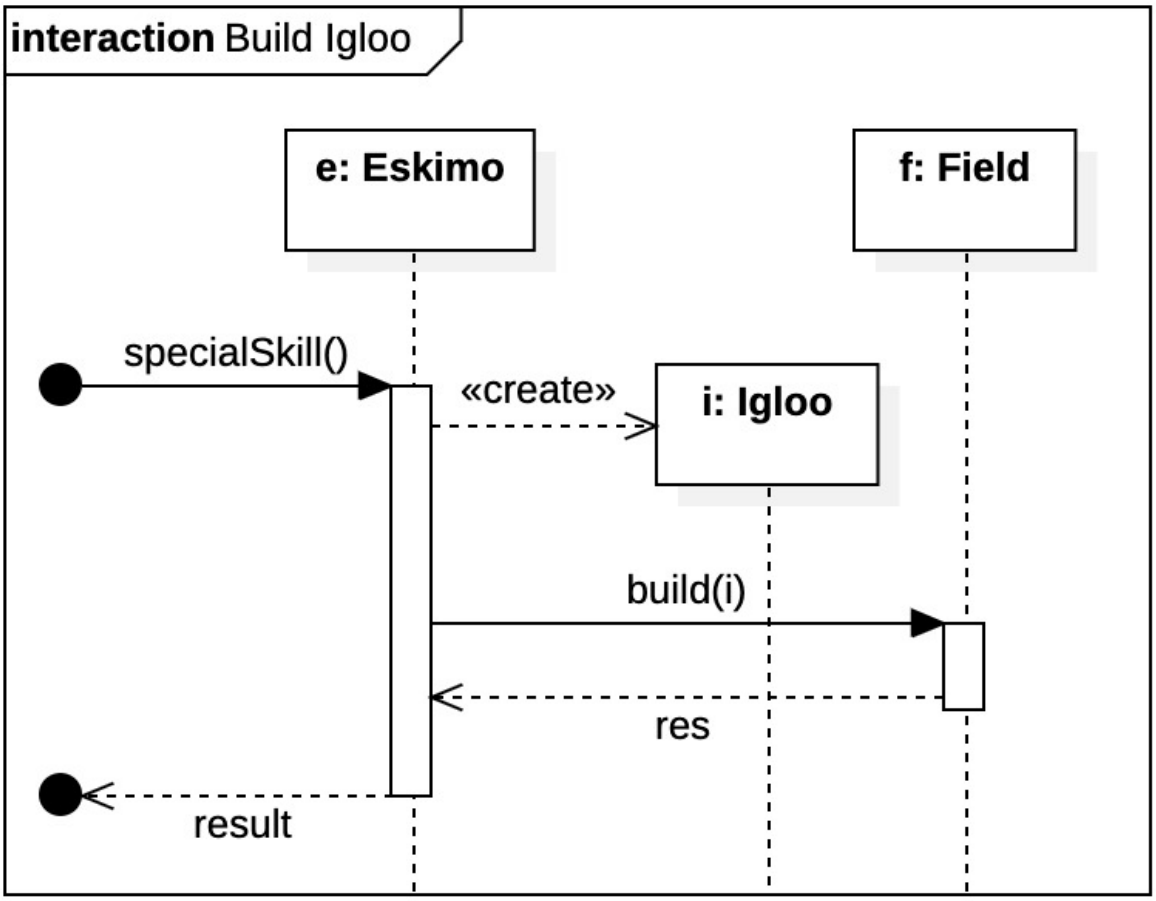
Az f példányt a clean(Field) függvény attribútumában megkapja a Shovel, tehát ismeri.

7.0.3.9 Clean with Breakable Shovel



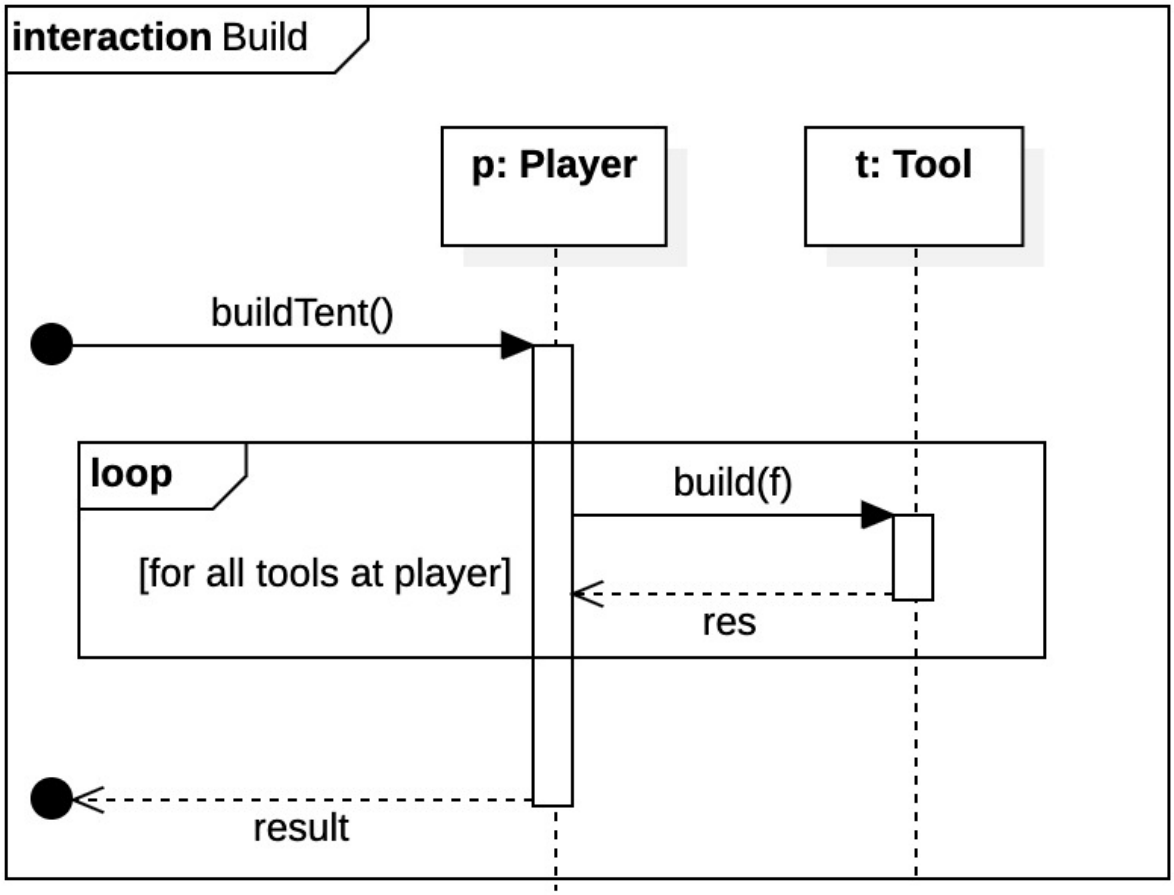
Az f példányt a clean(Field) függvény attribútumában megkapja a BreakableShovel, tehát ismeri.

7.0.3.10 Build Igloo



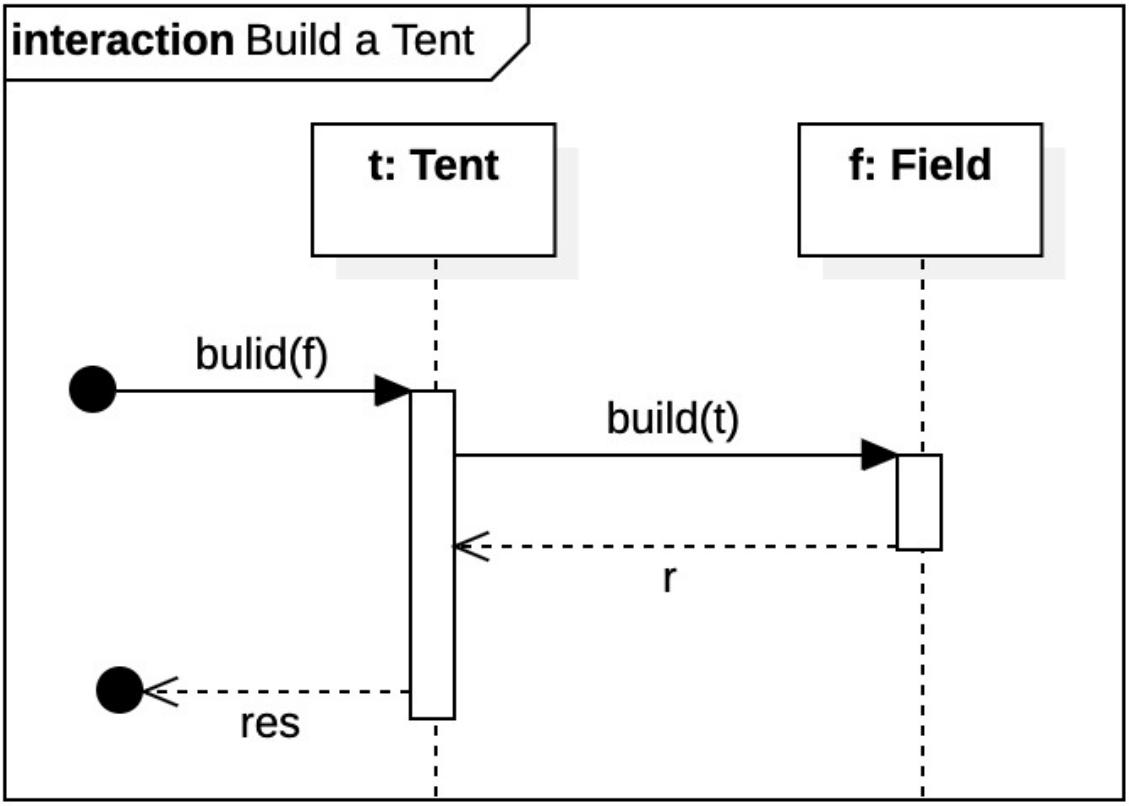
Az f példány megegyezik a Player osztály actualfield attribútumában tárolt példánnyal, tehát ismeri a Player.

7.0.3.11 Build



A t Tool-okat a játékos a tools attributumában tárolja, tehát ismeri azokat

7.0.3.12 Build a Tent



Az f példányt a build(Field) függvény attribútumában megkapja a Tent, tehát ismeri.

7.1 *Prototípus interface-definíciója*

7.1.1 Az interfész általános leírása

A prototípus interfészt parancssorból vezérelhetjük. A program a szabványos bemenetről fogad parancsokat a bemeneti nyelv formátumában és a szabványos kimenetre írja a kimenetét. A parancsok megadásának egyszerűsítése érdekében a program tud fájlból kapott inputok alapján működni. A tesztelés során az egyes tesztesetek bemenetét is így fogja kapni. A kimenet szintén átirányítható fájlba. A teszteset a bemenetből és az elvárt kimenetből fog állni.

7.1.2 Bemeneti nyelv

addplayer

Leírás: Létrehoz egy játékost a megadott színnel és típussal. Fontos, hogy két azonos színű playert nem adhatnak meg!
Szintaxis:
addplayer <szín> <típus>

A szín az alábbiak lehetnek:
p: lila
b: kék
g: zöld
y: citromsárga
o: narancssárga
r: piros
A típus kétféle lehet:
es: eszkimó
ex: sarkkutató

Példa:
addplayer p es

parsemap

Leírás: A paraméterben megadott sor- és oszlopszámú, illetve elrendezésű pálya mezőit létrehozza, illetve összeköti a szomszédaival.
Szintaxis:
parsemap <sorok száma> <oszlopok száma>
<f_1_1><f_1_2>...<f_1_n>
<f_2_1><f_2_2>...<f_2_n>
...
<f_m_1><f_m_2>...<f_m_n>

A sorok közben nincsen szóköz, a sorokat enter választja el. Az fontos hogy a bal felső(f_1_1) mező mindenképpen stabil mező lesz, ezért ha rosszul adják meg, akkor felülírja a program.
Számokkal adhatóak meg a mezők, az alábbi módon:
0: lyuk
1-6: a szám a teherbírást fogja megadni, amennyiben a szám egyenlő a játékosok számával, akkor a mező stabil jégtábla lesz, amennyiben kisebb az esetben instabil jégtábla lesz.

Példa:
parsemap 3 4
6236
1420
0305

parseitem

Leírás: A parseitem utasítást enter követi majd utána a parsemap-nak megfelelő számú sornak és oszlopnak kell lennie. Ahol a parsemapen 0- szerepel, tehát lyuk van, ott nem lehet item elásva, amennyiben lyukban lenne item failelni fog a teszt

Szintaxis:

parseitem
<i_1_1><i_1_2>...<i_1_n>
<i_2_1><i_2_2>...<i_2_n>
...
<i_m_1><i_m_2>...<i_m_n>

A sorok közben nincsen szóköz, a sorokat enter választja el. Valamely itemeknél számbeli korlátozások lehetnek. Betűkkel adhatóak meg az itemek, az alábbi módon:

g: Jelzőrakéta egyik része. Ebből egy pályán pontosan 3db-t kell elhelyezni!
f: Étel
s: Törhetetlen ásó
b: Törhető ásó
r: Kötél
d: Búvárruha
t: Sátor
n: üres mező

Példa:

parseitem
nnfg
sgbn
ntng

move

Leírás: A megadott játékost, a megadott irányba lépteti. A játékost a színével adhatja meg.

Szintaxis:

move <játékos színe> <irány>

A szín a létrehozásnál megfelelő szín.
Az irány pedig négy féle lehet:

u: felfelé
r: jobbra
d: lefelé
l: balra

Példa:

move p l

clean

Leírás: A megadott játékos ellapátolja a havat arról a mezőről amin van. A játékost a színével adhatja meg. Amennyiben van ásója a játékosnak kettő egységnyi havat takarít el, egyébként meg egy egységnyit.

Szintaxis:

clean <játékos színe>

Példa:

clean p

pickup

Leírás: A megadott játékos felvesz egy tárgyat(étel is lehet), arról a mezőről amelyiken áll, abban az esetben, ha a mezőt nem borítja hó. A játékost a színével adhatja meg. Amennyiben van már a játékosnak olyan tárgya, abban az esetben nem veszi fel. Amennyiben maximális a játékos testhője nem veszi fel az ételt.
Szintaxis:
pickup <játékos színe>
Példa:
pickup p

buildigloo

Leírás: A megadott játékos(csak eszkimó lehet!) használja a speciális képességét, azaz arra a mezőre amin áll épít egy igloot, abban az esetben ha azon még nincsen igloo. Az eszkimót a színével adhatja meg.
Szintaxis:
buildigloo <játékos színe>
Példa:
buildigloo p

examine

Leírás: A megadott játékos(csak sarkkutató lehet!) használja a speciális képesség, azaz megvizsgálja egy mellette levő mezőnek a teherbírását. A sarkkutatót a színével adhatja meg, az irányok pedig a ‘move’ parancsnál leírtaknak megfelelőek, tehát u/r/d/l
Szintaxis:
examine <játékos színe> <irány>
Példa:
examine b r

buildtent

Leírás: A megadott játékos, amennyiben rendelkezik sátorral, arra a mezőre amin áll épít egy sátrat. A játékost a színével adhatja meg.
Szintaxis:
buildtent <játékos színe>
Példa:
buildtent b

assemble

Leírás: Amennyiben a játékosok egy mezőn állnak és rendelkeznek a jelzőrakéta mindhárom alkatrészével összeszerelik a jelzőrakétát.
Szintaxis:
assemble
Példa:
assemble

storm

Leírás: A játékban az egész táblán végigsöpör a vihar. A mezők közül véletlenszerűen növeli a hómennyiséget egy egységgel.

Opciók:

- Determinisztikus vihar

A felhasználó megadhatja melyik mezőket sújtsa a vihar. A számozás 0-tól kezdődik! A mezőket spacel válasszuk el.

Szintaxis:

storm -r <mező> <mező>...<mező>

Példa:

storm -r f_1_1 f_1_2 f_2_2

- Véletlenszerű vihar

A vihar rendeltetésszerűen fog mezőket sújtani és így megnövelni a eggyel a hómennyiségüket.

Szintaxis:

storm

bear

Leírás: A játékban a jegesmedve minden körben egyet lép véletlenszerű irányba. A determinisztikus működés miatt a meg lehet külön adni, hogy a medve milyen irányba lépjen.

Opciók:

- Determinisztikus medve

A felhasználó a feljebb leírtak alapján megadhat egy irányt, amerre a medve lépni fog. Az irány lehet: u/r/d/l

Szintaxis:

bear -r <irány>

Példa:

bear -r u

- Véletlenszerű medve

A medve rendeltetésszerűen fog egy random irányba lépni. Ez egy kapcsoló, tehát csak akkor lép, amikor lépnie kell, nem feltétlen a parancs kiadásakor.

Szintaxis:

bear r

Példa:

bear r

state

Leírás: A paraméterként kapott entitás tulajdonságait(attribútumait) kiírja a szabványos kimenetre.

Szintaxis:

state <entitásnév>

Példa:

state p

state f_1_3

7.1.3 Kimeneti nyelv

A state parancsot kivéve minden parancs csak hiba esetén ír ki valamit a szabványos kimenetre, ami egy tömör hibaüzenet.
Az alapesetekben a mintaértékek teljesen véletlenszerűek, csak szemléltetésképpen szerepelnek.

Alapesetek:
Eskimo:
color: p
heat: 5
work: 3
tools: rope, shovel
actualfield: f_2_3
Explorer:
color: g
heat: 4
work: 2
tools: divingsuit, tent
actualfield: f_1_3
IceField:
snow: 3
capacity: 5
protection: false
item: rope
players: p, g
polarbear: true
neighborUp: f_3_3
neighborRight: f_2_4
neighborDown: f_2_2
neighborLeft: f_1_3
Hole:
snow: 4
capacity: 0
players: p
polarbear: false
neighborUp: f_0_2
neighborRight: f_3_1
neighborDown: f_1_2
neighborLeft: f_1_1
Polarbear:
actualfield: f_1_2
Game:
flaregun: charge, gun
players: p, g, b

Törékeny ásó:
used: 2

7.2 Összes részletes use-case

Use-case neve	Add player
Rövid leírás	Játékost ad hozzá.
Aktorok	Controller
Forgatókönyv	1.Hozzáad egy játékost

Use-case neve	Parse map
Rövid leírás	A megadott pálya létrehozása.
Aktorok	Controller
Forgatókönyv	1. A mezők létrehozása 2. A mezők szomszédossági viszonyainak beállítása

Use-case neve	Parse item
Rövid leírás	A megadott itemek elhelyezése.
Aktorok	Controller
Forgatókönyv	1. Az itemek hozzáadása a megfelelő mezőkhöz

Use-case neve	Move
Rövid leírás	Egy játékos mozgatása.
Aktorok	Player
Forgatókönyv	1. Annak vizsgálata, hogy a kiválasztott irányba tenger van-e 2. Amennyiben nem tenger van, a játékos átléptetése a mezőre

Use-case neve	Clean the field
Rövid leírás	Azon mezőről egy vagy két egységnyi hó eltávolítása amin a játékos áll.
Aktorok	Player
Forgatókönyv	1. Egy egységnyi hó eltakarítása 2. Amennyiben rendelkezik ásóval(törékeny ásó is lehet) még egy egységnyi hó eltakarítása 3. Amennyiben törékeny ásót használt a player a használtsági mutatóját csökkenti eggyel

Use-case neve	Pick up an item
Rövid leírás	Egy tárgy felvétele.
Aktorok	Player
Forgatókönyv	1. Annak vizsgálata, hogy az mezőn nincsen hó és van felvehető item 2. Amennyiben a játékos még nem rendelkezik olyan itemmel(ételnél nem maximális a testhője) felveszi

Use-case neve	Build igloo
Rövid leírás	Eszkimó igloot épít.
Aktorok	Player
Forgatókönyv	1. Annak vizsgálata, hogy van-e már igloo vagy sátor a mezőn 2. Amennyiben nincs, akkor épít egyet

Use-case neve	Examine capacity
Rövid leírás	Sarkkutató teherbírást vizsgál.
Aktorok	Player
Forgatókönyv	1. Annak vizsgálata, hogy a kiválasztott irányba tenger van-e 2. Amennyiben nem tenger van, a teherbírás vizsgálata

Use-case neve	Build tent
Rövid leírás	Játékos sátrat épít.
Aktorok	Player
Forgatókönyv	1. Annak ellenőrzése, hogy a játékos rendelkezik-e ásóval 2. Amennyiben rendelkezik, annak a vizsgálata, hogy van-e már igloo vagy sátor a mezőn 3. Amennyiben nincs, akkor épít egyet

Use-case neve	Assemble the fire gun
Rövid leírás	A jelzőrakéta összeszerelése.
Aktorok	Player
Forgatókönyv	1. Annak vizsgálata, hogy mindhárom alkatrész kiásták 2. Amennyiben rendelkeznek mindhárom alkatrésszel rendelkeznek, annak vizsgálata, hogy egy mezőn állnak a játékosok 3. Amennyiben ez is teljesült a jelzőrakéta összeszerelése

Use-case neve	Storm comes to the fields
Rövid leírás	A vihar valamely mezőket újabb réteg hóval fedi el.
Aktorok	Controller
Forgatókönyv	1. A megadott/véletlenszerűen kiválasztott mezőket fedő hómennyiséget egy egységgel növeli 2. Azon játékosok testhőjének csökkentése, akik vihar sújtotta mezőn állnak, de a mező nem sátorral vagy iglooval védett

Use-case neve	Ice bear moves
Rövid leírás	A jegesmedve mozgatása.
Aktorok	Controller
Forgatókönyv	1. Annak vizsgálata, hogy a megadott/véletlenszerűen kiválasztott irányba tenger van-e 2. Amennyiben nem tenger, a jegesmedve mozgatása 3. Ha azon a mezőn, amire lépett a medve játékos van és nincs igloo a mezőn, akkor a játékos meghal és vége a játéknak

Use-case neve	Show state
Rövid leírás	Információ megtekintése az adott entitásról.
Aktorok	Controller, Player
Forgatókönyv	1. Információ kiírása 2. A játékos megnézi az információt

7.3 *Tesztelési terv*

Teszt-eset neve	**1. Játékos lyukra lép**
Rövid leírás	A játékos egy szomszédos lyukra lép és nem mentik ki akkor meghal.
Teszt célja	Teszteli, hogy ha egy játékos lyukra lép akkor vége a játéknak. (Hole, Game)

Teszt-eset neve	**2. Játékost kimentik**
Rövid leírás	A játékost kimentik miután lyukra esett
Teszt célja	Teszteli, hogy ha egy játékos lyukra lép, és van olyan szomszédos játékos aki ki tudja menteni akkor megmenekül. (Player, Rope, Hole)

Teszt-eset neve	**3. Játékos lyukra lép és úszik**
Rövid leírás	A játékos lyukra lép, és rendelkezik búvárruhával akkor kiúszik.
Teszt célja	Teszteli, hogy ha egy játékos lyukra lép akkor kiúszik egy szomszédos jégtáblára. (Player, Food)

Teszt-eset neve	**4. Játékos stabil jégmezőre lép**
Rövid leírás	A játékos egy szomszédos jégtáblára lép.
Teszt célja	Teszteli, hogy ha egy játékos szomszédos jégtáblára lép. (Player, IceField)

Teszt-eset neve	5. Játékos instabil jégmezőre lép
Rövid leírás	A játékos egy szomszédos jégtáblára lép és annak a kapacitása már a maximumon van akkor felborul, és vége a játéknak.
Teszt célja	Teszteli, hogy ha egy játékos instabil jégtáblára lép és maximális a teherbírása akkor felborul és vége a játéknak. (Player, IceField, Game)

Teszt-eset neve	6. Vihar iglu vagy sátor nélkül
Rövid leírás	Ha jön a vihar megadott mezőkre akkor azokon növekszik a hómennyiség 1-el. Ha az érintett mezőkön áll játékos és nincs iglu akkor a testhőjük csökken 1 egységgel.
Teszt célja	Teszteli, hogy növekszik-e a hómennyiség a vihar által érintett mezőkön, valamint a védtelen játékosok testhője is csökken 1 egységgel. (Storm, Field, Player)

Teszt-eset neve	7. Vihar igluval vagy sátorral
Rövid leírás	Ha jön a vihar megadott mezőkre akkor azokon növekszik a hómennyiség 1-el. Ha az érintett mezőkön áll játékos igluban akkor a testhőjük nem csökken
Teszt célja	Teszteli, hogy növekszik-e a hómennyiség a vihar által érintett mezőkön, valamint az igluban vagy sátorban lévő játékosok testhője nem változik. (Storm, Field, Player)

Teszt-eset neve	8. Játékos felvesz egy tárgyat tiszta jégtábláról
Rövid leírás	A játékos felvesz egy tiszta jégtábláról egy tárgyat
Teszt célja	Teszteli, hogy ha tiszta egy jégtábla akkor fel tud venni a játékos egy tárgyat. (Player, Tool implementálói)

Teszt-eset neve	9. Játékos felvesz egy tárgyat havas jégtábláról
Rövid leírás	A játékos megpróbál felvenni tárgyat egy havas jégtábláról
Teszt célja	Teszteli, hogy ha egy játékos megpróbál tárgyat felvenni egy havas tábláról (nem tud felvenni)

Teszt-eset neve	10. Játékos megpróbál felvenni egy birtokolt tárgyat
Rövid leírás	A játékos megpróbál felvenni egy olyan tárgyat amilyen tárgya már van.
Teszt célja	Teszteli, hogy egy játékos felvesz egy olyan tárgyat amilyennel már rendelkezik. (nem tudja)

Teszt-eset neve	11. Játékos felvesz ételt
Rövid leírás	A játékos felvesz ételt és az növeli a testhőjét
Teszt célja	Teszteli, hogy ha egy játékos felvesz ételt az növeli a testhőjét. (Player, Food)

Teszt-eset neve	12. Játékos megpróbál ételt felvenni, de maximumon van a testhője
Rövid leírás	A játékos megpróbál ételt felvenni
Teszt célja	Teszteli, hogy ha egy játékos felvesz ételt maximális testhővel. (nem tudja felvenni)

Teszt-eset neve	13. Játékos kézzel ás
Rövid leírás	A játékos ellapátol 1 egység havat.
Teszt célja	Teszteli, hogy ha egy játékos kézzel ás akkor a jégtáblán lévő hómennyiség csökken 1 egységgel. (Player, Field)

Teszt-eset neve	14. Játékos ásóval ás
Rövid leírás	A játékos ellapátol 2 egység havat
Teszt célja	Teszteli, hogy ha egy játékos ásóval ás akkor a jégtáblán lévő hómennyiség 2 egységgel csökken. (Player, Shovel, Field)

Teszt-eset neve	15. Játékos törékeny ásóval ás
Rövid leírás	A játékos ellapátol 2 egység havat és 3 ásás után összetörik.
Teszt célja	Teszteli, hogy ha egy játékos ásóval ás akkor a jégtáblán lévő hómennyiség 2 egységgel csökken és az ásó használtsága nő, 3 ásás után eltűnik. (Player, Field, Breakable Shovel)

Teszt-eset neve	16. Eszkimó iglut épít
Rövid leírás	Az eszkimó iglut épít egy jégtáblára, ha még nincs rajta.
Teszt célja	Teszteli, hogy ha egy eszkimó épít egy iglut a jégtáblára ahol még nincs. (Eskimo, Field, Igloo)

Teszt-eset neve	17. Játékos sátrat épít
Rövid leírás	A játékos rendelkezik sátorral és épít egy sátrat a jégtáblára és 1 kör után eltűnik,.
Teszt célja	Teszteli, hogy a játékos ha rendelkezik sátorral és egy jégtáblán még nincs sátor vagy iglu akkor épít és az eltűnik 1 kör után.. (Player, Field, Tent)

Teszt-eset neve	18. Jegesmedve iglu nélküli mezőre lép
Rövid leírás	A jegesmedve olyan mezőre lép ahol áll egy játékos, ekkor vége a játéknak.
Teszt célja	Teszteli, hogy ha a jegesmedve olyan mezőre lép ahol áll játékos akkor vége a játéknak. (Player, Game, PolarBear)

Teszt-eset neve	19. Jegesmedve igluval rendelkező mezőre lép
Rövid leírás	A jegesmedve olyan mezőre lép ahol áll játékos, de iglu is van a mezőn akkor a játékos megmenekül.
Teszt célja	Teszteli, hogy ha a jegesmedve olyan mezőre lép ahol van játékos, de igluban van akkor nem bántja a medve. (Player, Field, PolarBear, Igloo)

Teszt-eset neve	20. Játékosok összegyűjtik a jelzőrakéta elemeit
Rövid leírás	A játékosok összegyűjtik a jelzőrakéta elemeit, és ugyanarra a mezőre mennek akkor megnyerik a játékot.
Teszt célja	Teszteli, hogy ha a játékosok összegyújtötték az összes elemét a jelzőrakétának és ugyanazon a mezőn vannak akkor megnyerték a játékot. (Player, Field, Igloo)

Teszt-eset neve	21. Sarkkutató megvizsgálja a szomszédos mező kapacitását
Rövid leírás	A sarkkutató egy szomszédos mezőről lekérdezi annak teherbírását
Teszt célja	Teszteli, hogy egy sarkkutató lekérdezi egy szomszédos mező teherbírását. (Explorer, Field)

Teszt-eset neve	22. Játékos munkája csökken
Rövid leírás	A játékosnak csökken 1 egységgel az elvégezhető munkáinak száma az alábbi tevékenységekre: lépés, ásás, speciális képesség használata, tárgy felvétel.
Teszt célja	Teszteli hogy a játékos elvégezhető munkáinak száma csökken-e valamely előbb felsorolt tevékenység hatására (Player)

Megjegyzés:

- Ahol játékos szerepel a tesztesetekben az közösen eszkimót vagy sarkkutatót is jelent, mert azok a cselekmények nem szereplőspecifikusak.
- A 1-21 tesztesetekben ha valamely tevékenység csökkentené a játékos munkavégzéseit az nincs feltüntetve, hogy a konkrét hatásra helyeződjön a fókusz

7.4 *Tesztelést támogató segéd- és fordítóprogramok specifikálása*

A prototípus tesztelését egy PowerShell script fogja megkönnyíteni. A scriptet lehet használni többféleképpen: egy teszteset futtatása esetén a teszteset fájljait paraméterül adva, míg minden teszteset lefuttatásához egy *-ot kell paraméterül adnunk. A teszteset egy *teszteset.in* és *testeset.out* fájlból állnak. A *.in* fájl a bemeneti nyelv formátumában parancsokat tartalmaz amiket az adott tesztesethez ki kell adni. A *.out* fájl az elvárt kimenetet tartalmazza amit a script összehasonlít a kapott kimenettel és ez alapján eldönti a teszt sikerességét. Ha sikeres a teszt akkor ezt egy üzenettel kiírja, ha sikertelen akkor a kimenetek közötti eltérést kiírja a képernyőre.

Napló

Kezdet	Időtartam	Résztvevők	Leírás
2020.04.02. 13:30	1,5 óra	Fábián Gutai Ilosvay Szabó K. Szabó M.	Értekezlet. A feladatok felosztása, és a bevezetett módosítások átbeszélése, értelmezése.
2020.04.04. 13:00	2 óra	Fábián	Az osztálydiagram módosítása, és a 7.0.3.1, 7.0.3.2, 7.0.3.3, szekvenciák elkészítése
2020.04.04. 15:00	1 óra	Fábián	A 7.0.3.6, 7.0.3.7, 7.0.3.8, 7.0.3.9, szekvenciák elkészítése
2020.04.04. 17:00	2 óra	Fábián	A 7.0.3.4, 7.0.3.5, 7.0.3.10, 7.0.3.11, 7.0.3.12 szekvenciák elkészítése
2020.04.04. 20:00	3 óra	Gutai	Use-casek megírása
2020.04.04. 22:00	4 óra	Ilosvay	7.0.2 megírása
2020.04.04. 20:00	4 óra	Szabó K	7.1.2 és 7.1.3 megírása
2020.04.04. 20:00	4 óra	Szabó M	7.3 megírása
2020.04.05. 00:30	1,5 óra	Szabó K	7.2 újraírása
2020.04.05. 00:30	1,5 óra	Szabó M	7.4 és 7.1.1 megírása